A programozás alapjai 2.

Házi Feladat Dokumentáció

**File titkosító program**

Készítette: Albrecht Ádám, O9E6U1

adam.albrecht@edu.bme.hu

Készítés féléve: 2024/25/2

Tartalomjegyzék

[Felhasználói dokumentáció 3](#_Toc198411927)

[Elvárt bemenet 3](#_Toc198411928)

[Elvárt kimenet 3](#_Toc198411929)

[Példa működés 3](#_Toc198411930)

[Osztályok statikus leírása 3](#_Toc198411931)

[Node 3](#_Toc198411932)

[Felelőssége 3](#_Toc198411933)

[Attribútumok 3](#_Toc198411934)

[Metódusok 4](#_Toc198411935)

[List 4](#_Toc198411936)

[Felelőssége 4](#_Toc198411937)

[Attribútumok 4](#_Toc198411938)

[Metódusok 4](#_Toc198411939)

[FileManager 5](#_Toc198411940)

[Felelőssége 5](#_Toc198411941)

[Metódusok 5](#_Toc198411942)

[MenuRenderer 5](#_Toc198411943)

[Felelőssége 5](#_Toc198411944)

[Metódusok 5](#_Toc198411945)

[MenuSystem 5](#_Toc198411946)

[Felelőssége 5](#_Toc198411947)

[Attribútumok 5](#_Toc198411948)

[Metódusok 5](#_Toc198411949)

[UserInputHandler 6](#_Toc198411950)

[Felelőssége 6](#_Toc198411951)

[Metódusok 6](#_Toc198411952)

[UML osztálydiagram 7](#_Toc198411953)

[Összegzés 8](#_Toc198411954)

[Mit sikerült és mit nem sikerült megvalósítani a specifikációból? 8](#_Toc198411955)

[Mit tanultál a megvalósítás során? 8](#_Toc198411956)

[Továbbfejlesztési lehetőségek 8](#_Toc198411957)

[Képernyőképek a futó alkalmazásról 9](#_Toc198411958)

# Felhasználói dokumentáció

### Elvárt bemenet

A program egy menürendszerrel rendelkezik. Az egyes menüpontokat a megfelelő sorszámok beírásával érhetjük el. Első lépésként be kell olvasnunk egy bemeneti fájlt (1. Load Words from File), aminek a szövegét kódolni szeretnénk. Itt egy olyan szöveges fájl nevét kell megadnunk, ami a program könyvtárában található, illetve ékezetek nélküli szöveget tartalmaz. Az írásjelek nem kerülnek titkosításra.

### Elvárt kimenet

A program menüjéből választhatunk különböző funkciókból:

2. Save Words to File: miután megadtunk egy nekünk szimpatikus nevet a kimeneti fájlnak, generálódik egy fájl, amely tartalmazza a szavak egyedi kódjait a szavak előfordulási számát és a szavakat magukat egy táblázatban.

3. Encode Loaded File: miután megadtunk egy nekünk szimpatikus nevet a kimeneti fájlnak, generálódik egy fájl, amely tartalmazza az eredeti szöveg kódolt változatát, ahol a szavakat az egyedi kódok helyettesítik.

4. Visualize Word List: vizualizálja a láncolt listát, amiben a program a szavakat eltárolta, és ezt terminálra kiírja

5. Search for Word Code: miután megadunk egy szót, a program kiírja a hozzá rendelt azonosítót. Ha a szó nincs a listában, akkor ezt is kiírja.

### Példa működés

Olvassuk be a mellékelt „feladat.be” fájlt az első menüpont segítségével. Ezután a kettes menüponttal mentsük el a generált kódtáblát a „feladat.ki” fájlba, illetve a hármas menüponttal kódoljuk a bemeneti fájlunkat és mentsük el „kodolt.ki” néven. A négyes menüponttal kipróbálhatjuk a lista vizualizálását, ebből a módból ENTER lenyomásával léphetünk ki. Az ötös menüponttal keressünk rá a bemeneti fájl egyik szavára, hogy megkapjuk milyen kódot rendelt hozzá a programunk. Amennyiben szeretnénk a programot egy másik bemeneti fájllal is kipróbálni, a hatos menüponttal töröljük először a betöltött listát. Ezután az egyes menüpont ismételt használatával jöhet az új fájl. A programot bezárni a hetes menüponttal tudjuk.

# Osztályok statikus leírása

## Node

### Felelőssége

A Node osztály a láncolt lista egy elemét valósítja meg. Egy karaktert tárol, valamint mutatókat tartalmaz a lefele (child) és oldalirányban (sibling) következő node-ra. Külön mezőkben tárolja, hogy egy adott szó hányszor fordult elő, illetve milyen kód van hozzárendelve, ha a node egy szó végét jelöli.

### Attribútumok

#### Privát

* char letter; // Megadja, hogy az adott node milyen karaktert tárol.
* Node\* down; // Pointer a lefele következő node-ra, amely a következő karaktert tárolja a szóban.
* Node\* next; // Pointer a következő, azonos szinten lévő node-ra, amely más szót vagy szóágat kezd ugyanazon karakterpozíción.
* int count; // Megadja, hogy az adott szó hányszor fordul elő a kódolt szövegben. Csak akkor nem nulla, ha a node a szó végét jelöli.
* int code; // Az adott szóhoz tartozó kódot tárolja. Csak akkor nem nulla, ha a node egy szó végét jelöli.

### Metódusok

#### Publikus

* Node(char ch); // Konstruktor: beállítja a letter változót és minden pointert, számlálót alaphelyzetbe állít.
* ~Node(); // Destruktor: default, a node felszabadítását végzi.
* char getLetter() const; // Visszaadja a tárolt karaktert.
* Node\* getDown() const; // Visszaadja a lefele mutató pointert
* Node\* getNext() const; // Visszaadja az oldalra mutató pointert
* int getCount() const; // Visszaadja a count változó értékét, vagyis a szó előfordulásszámát.
* int getCode() const; // Visszaadja a code változó értékét, azaz a szóhoz rendelt kódot.
* Node\*& getDownRef(); // Referenciát ad vissza a lefele mutató pointerhez
* Node\*& getNextRef(); // Referenciát ad vissza az oldalra mutató pointerhez
* void setDown(Node\* ptr); // Beállítja a lefele mutató pointert.
* void setNext(Node\* ptr); // Beállítja az oldalra mutató pointert.
* void incrementCount(); // Eggyel növeli a count változót, ezzel követi a szó előfordulásait.
* void setCode(int c); // Beállítja a code értékét, vagyis a szóhoz tartozó kódot.

## List

### Felelőssége

Ez a class kezeli a láncolt listát, amelyben a szavakat karakterenként, láncolt listában tárolja, és minden szóhoz egyedi kódot rendel. Gondoskodik a szavak beszúrásáról, törléséről, kereséséről, kódolásáról, fájlba írásáról és vizualizációjáról.

### Attribútumok

#### Privát

* Node\* root; // Egy pointer a lista gyökér nodejára
* int nextCode; // Tárolja, hogy mi lesz a következő kód, amit hozzárendel egy új szóhoz

### Metódusok

#### Privát

* void printWords(Node\* node, std::string& path, std::ofstream& out) const; // Rekurzívan kiírja a fába épített szavakat, és azok kódjait, előfordulásszámát.
* void freeNodes(Node\* node); // Rekurzívan felszabadítja a dinamikusan foglalt node-okat, memóriaszivárgás elkerülésére.
* void visualize(Node\* node, const std::string& prefix, bool isLast) const; // Rekurzívan megjeleníti a fa struktúráját szövegesen, vizualizációs célból.

#### Publikus

* List(); // Konstruktor: inicializálja a root pointert nullptr-ra, előkészíti a listát használatra.
* ~List(); // Destruktor: felszabadítja a listához tartozó összes node-ot.
* void insertWord(const std::string& word); // Egy szót karakterenként beszúr a listába, ha kell, új ágakat hoz létre, és beállítja a kódot.
* int getWordCode(const std::string& word) const; // Visszaadja a megadott szóhoz tartozó kódot, vagy 0-t ha nem található.
* void encodeFile(const std::string& inputfile, const std::string& outfile) const; // Egy bemeneti fájl szavait kódokra cseréli, és kiírja egy kimeneti fájlba.
* friend std::ofstream& operator<<(std::ofstream& out, const List& list); // Kiírja a listát (szavak, kódok, előfordulások) egy fájlba.
* friend std::ifstream& operator>>(std::ifstream& in, List& list); // Betölti a szavakat egy fájlból, és beilleszti őket a listába.
* void visualize() const; // Megjeleníti a listastruktúráját, vizuális, könnyebb átláthatóság érdekében.

## FileManager

### Felelőssége

Ez a class intézi az összes fájlkezeléssel kapcsolatos dolgot.

### Metódusok

#### Publikus

* static bool loadWords(const string& fileName, List& list); // Betölti a szavakat egy fájlból, és beilleszti őket a listába.
* static bool saveWords(const string& fileName, const List& list); // Kiírja a lista szavait, kódjait és előfordulásait egy fájlba.
* static bool encodeFile(const string& inputFileName, const string& outputFileName, const List& list); // Egy teljes szöveges fájlt kódol át a szótár alapján.

## MenuRenderer

### Felelőssége

Ez az osztály felelős a felhasználói menü szöveges megjelenítéséért, dinamikusan alkalmazkodva a betöltött fájl állapotához.

### Metódusok

#### Publikus

* static void displayMenu(bool isFileLoaded, const string& loadedFileName); // Megjeleníti a főmenüt, kiemelve a betöltés állapotát és a lehetséges opciókat.

### MenuSystem

### Felelőssége

A MenuSystem osztály vezérli a teljes felhasználói interakciós folyamatot: menük megjelenítése, felhasználói választás kezelése, a megfelelő műveletek indítása (fájlbetöltés, kódolás, mentés, keresés, stb).

### Attribútumok

#### Privát

* List list; // A láncolt listát tárolja.
* bool isFileLoaded; // Jelzi, hogy be van-e töltve szótárfájl.
* string loadedFileName; // A betöltött fájl neve.

### Metódusok

#### Privát

* void handleUserChoice(int choice); // Feldolgozza a felhasználó menüválasztását, és végrehajtja a megfelelő műveletet.

#### Publikus

* void run(); // Elindítja a menürendszert és lebonyolítja a fő programfolyamot.
* static void clearScreen(); // Törli a konzolt (képernyőt)

## UserInputHandler

### Felelőssége

Ez az osztály felelős a felhasználói inputok biztonságos és ellenőrzött bekéréséért, valamint validálásáért.

### Metódusok

#### Publikus

* static int getValidatedChoice(); // Bekéri és ellenőrzi a felhasználó menüválasztását, csak számot enged.
* static string getValidatedString(const string& prompt); // Bekér egy tetszőleges stringet a felhasználótól, a megadott prompttal.

# UML osztálydiagram

A képen szöveg, képernyőkép, szoftver, Weblap látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

# Összegzés

## Mit sikerült és mit nem sikerült megvalósítani a specifikációból?

Nagyjából ugyanazt valósítottam meg, mint amit a specifikációban leírtam. A menürendszerem egyel komplikáltabb lett, mint, ahogy eredetileg terveztem.

## Mit tanultál a megvalósítás során?

Megtanultam, hogy hogy lehet jól szétbontani egy programot classokra. Illetve a láncolt listák kezeléséről és bejárásáról elég sok mindent.

## Továbbfejlesztési lehetőségek

Lehetne hozzáadni dekódolást, különböző statisztikai kiírásokat. Esetleg tömörítő programmá is tovább lehetne fejleszteni.

# Képernyőképek a futó alkalmazásról

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, elektronika, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.